

Fast Vertex Guarding for Polygons

James King

Department of Physics
University of Oxford

Abstract. For a polygon P with n vertices, the vertex guarding problem asks for the minimum subset G of P 's vertices such that every point in P is seen by at least one point in G . This problem is NP-complete and APX-hard. The first approximation algorithm (Ghosh, 1987) involves decomposing P into $\mathcal{O}(n^4)$ cells that are equivalence classes for visibility from the vertices of P . This discretized problem can then be treated as an instance of set cover and solved in $\mathcal{O}(n^5)$ time with a greedy $\mathcal{O}(\log n)$ -approximation algorithm. Ghosh (2010) recently revisited the algorithm, noting that minimum visibility decompositions for simple polygons (Bose *et al.*, 2000) have only $\mathcal{O}(n^3)$ cells, improving the running time of the algorithm to $\mathcal{O}(n^4)$ for simple polygons.

In this paper we show that, since minimum visibility decompositions for simple polygons have only $\mathcal{O}(n^2)$ cells of *minimal* visibility (Bose *et al.*, 2000), the running time of the algorithm can be further improved to $\mathcal{O}(n^3)$. This result was obtained independently by Jang and Kwon (2011). We extend the result of Bose *et al.* to polygons with holes, showing that a minimum visibility decomposition of a polygon with h holes has only $\mathcal{O}((h+1)n^3)$ cells and only $\mathcal{O}((h+1)^2n^2)$ cells of minimal visibility. We exploit this result to obtain a faster algorithm for vertex guarding polygons with holes. We then show that, in the same time complexity, we can attain approximation factors of $\mathcal{O}(\log \log \text{OPT})$ for simple polygons and $\mathcal{O}((1 + \log(h+1)) \log \text{OPT})$ for polygons with holes.

1 Introduction

Art gallery problems, *i.e.*, polygon guarding problems, are motivated by the question, “How many security cameras are required to guard an art gallery?” The art gallery is modeled as a connected polygon P . A camera, which we henceforth call a *guard*, is modeled as a point in the polygon, and we say that a guard g *sees* a point q in the polygon if the line segment \overline{gq} is contained in P . The visibility polygon of a point p , denoted $\text{Vis}(p)$, is the set of points in P that see p . We call a set G of points a *guarding set* if every point in P is seen by some $g \in G$, *i.e.*, if $\bigcup_{g \in G} \text{Vis}(g) = P$. Let $V(P)$ denote the vertex set of P and let ∂P denote the boundary of P . We assume that P is closed and non-degenerate so that $V(P) \subset \partial P \subset P$.

We consider the minimization problem that asks, given an input polygon P with n vertices, for a minimum guarding set for P . Variants of this problem typically differ based on what points in P must be guarded and where guards can be placed, as well as whether P is simple or contains holes. Typically we want to guard either P or ∂P , and our set of potential guards is typically $V(P)$ (vertex guards), ∂P (perimeter guards), or P (point guards). This paper concerns the variant in which we must guard all of P from vertices of P . For results on art gallery problems not related to minimization problems we direct the reader to O’Rourke’s book [22], which is available for free online.

1.1 Related Work

Hardness Results The problem was proved to be NP-complete for polygons with holes by O’Rourke and Supowit [23]. For guarding simple polygons it was proved to be NP-complete for vertex guards by Lee and Lin [21]; their proof was generalized to work for point guards by Aggarwal [1]. This raises the question of approximability. There are two major hardness results. First, for guarding simple polygons, Eidenbenz [10] proved that the problem is APX-complete, meaning that we cannot do better than a constant-factor approximation algorithm in polynomial time unless $P = NP$. Subsequently, for guarding polygons with holes, Eidenbenz *et al.* [11] proved that the minimization problem is as hard to approximate as set cover in general if there is no restriction on the number of holes. It therefore follows from results about the inapproximability of set cover [25,13,2] that, for polygons with an unbounded number of holes, it is NP-hard to find a guarding set of size $o(\log n)$. These hardness results hold whether we are dealing with vertex guards, perimeter guards, or point guards.

Approximation Algorithms Ghosh [14] provided an $O(\log n)$ -approximation algorithm for guarding polygons with or without holes with vertex guards. His algorithm decomposes the input polygon into a polynomial number of cells such that each point in a given cell is seen by the same set of vertices. This discretization allows the guarding problem to be treated as an instance of set cover and solved using general techniques. In fact, applying methods for set cover developed after Ghosh’s algorithm, it is easy to obtain an approximation factor of

$O(\log \text{OPT})$ for vertex guarding simple polygons or $O(\log h \log \text{OPT})$ for vertex guarding a polygon with h holes.

When considering point guards or perimeter guards, discretization is far more complicated since two distinct points will not typically be seen by the same set of potential guards even if they are very close to each other. Deshpande *et al.* [8] obtain an approximation factor of $O(\log \text{OPT})$ for point guards or perimeter guards by developing a sophisticated discretization method that runs in pseudopolynomial time¹. Efrat and Har-Peled [9] provided a randomized algorithm with the same approximation ratio that runs in fully polynomial expected time; their discretization technique involves only considering guards that lie on the points of a very fine grid.

1.2 Range Spaces and Discretization

Guarding problems can naturally be expressed as instances of set cover or hitting set. We wish to model an instance of a guarding problem as an instance of hitting set on a range space $\mathcal{S} = (X, \mathcal{R})$, constructed as follows. X is equal to the set S_G of potential guard locations. For each point p that needs to be guarded, R_p is the set of potential guards that see p . Now $\mathcal{R} = \{R_p : p \in S_T\}$, where S_T is the set of points that must be guarded. For the vertex guarding problem, $S_G = V(P)$ and $S_T = P$.

We assume S_G is finite; in our case $|S_G| = n$. If S_T is *not* finite, *e.g.*, when $S_T = P$, we need to discretize it. The goal of discretization is to find a finite representative subset $S'_T \subset S_T$ such that any subset of S_G that guards S'_T also guards S_T . With such a set we are able to forget about S_T and focus on the finite range space $(S_G, \{R_p : p \in S'_T\})$ induced by S_G and S'_T .

We consider general techniques for solving hitting set for finite range spaces. For a finite range space $\mathcal{S} = (X, \mathcal{R})$ the time complexity typically depends on $|X|$ (*i.e.*, the number of elements) and $|\mathcal{R}|$ (*i.e.*, the number of ranges).

1.3 Visibility Decompositions for Polygons

For points $p, q \in S_T$, we say that p and q are equivalent if and only if $R_p = R_q$. In a decomposition of a polygon into cells, we say that a cell is an *equivalence cell* if all points are equivalent. A natural discretization strategy is to partition S_T into a finite number of sets that are closed under equivalence, and then to build a subset S'_T by taking one representative point from each set. A subset of S_G guards S_T if and only if it guards S'_T .

Ghosh [14] did this for the vertex guarding problem in which $S_G = V$ and $S_T = P$. His algorithm decomposes the input polygon into a polynomial number of cells such that each point in a given cell is seen by the same set of vertices. For two distinct vertices that see each other, consider the line through them. The set of all such lines decomposes P into a number of equivalence cells (see Figure

¹ It is a pseudopolynomial-time algorithm in that its running time may be linear in the ratio between the longest and shortest distances between two vertices.

1). Using a simple inequality for general line arrangements it can be seen that the number of cells is $\mathcal{O}(n^4)$. Ghosh originally used his discretization technique, along with the greedy set cover approximation algorithm (see, *e.g.*, [29, pp. 16–19]), to provide a $\mathcal{O}(\log n)$ -approximation algorithm. This decomposition can be generalized to work for any finite set S_G of potential guards—simply shoot a ray from every point $p \in S_G$ through every vertex seen by p . In each of the $\mathcal{O}(|S_G|^2 n^2)$ resulting cells, any two points see the same subset of S_G .

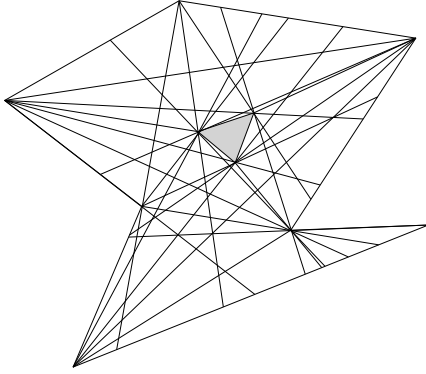


Fig. 1. A polygon decomposed into $\mathcal{O}(n^4)$ cells by cutting along any line passing through two vertices that see each other. The shaded region is a hole.

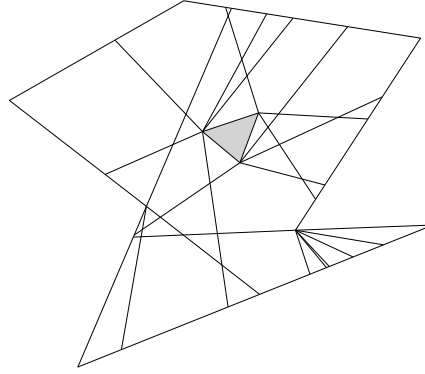


Fig. 2. A polygon decomposed into the $\mathcal{O}(n^3)$ cells of its minimum visibility decomposition $\mathcal{D}_V(P, V(P))$. The shaded region is a hole.

Bose *et al.* [5] introduced a minimum decomposition with fewer cells (some of their results were obtained independently by Guibas *et al.* [16], but we focus on the results as stated and proved by Bose *et al.*). Let the *visibility decomposition*, denoted $\mathcal{D}_V(P, S_G)$, be the minimum decomposition of a polygon into equivalence cells with regard to S_G . It is minimum in that the union of cell boundaries in this decomposition is exactly equal to $\bigcup_{p \in S_G} \partial(\text{Vis}(p))$. The decomposition can be constructed as follows. For a point $p \in S_G$ that sees a vertex v , instead of cutting along the entire ray shot from p through v , we leave the line segment \overline{pv} and cut only from v until we hit ∂P (see Figure 2). Bose *et al.* call such a cut a *window of point p* and they note that the boundary of $\text{Vis}(p)$ consists only of windows of p and parts of ∂P . They proved an upper bound of $\mathcal{O}(n^3)$ for the number of cells in $\mathcal{D}_V(P, V(P))$ for a simple polygon P .

Ghosh [15] recently revisited his algorithm, using these minimum visibility decompositions to improve the running time. His updated algorithm guarantees an approximation factor of $\mathcal{O}(\log n)$ and runs in $\mathcal{O}(n^4)$ for simple polygons and $\mathcal{O}(n^5)$ for polygons with holes.

1.4 Novel Contributions

In this paper we exploit another result of Bose *et al.* [5], namely the fact that, while minimum visibility decompositions for simple polygons can have $\Theta(n^3)$ cells, they have only $\mathcal{O}(n^2)$ cells of *minimal* visibility. We use this to further improve the running time of Ghosh’s algorithm from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^3)$.

We then extend the result of Bose *et al.* to polygons with h holes, parameterizing bounds not only by n but also by h . We show that a minimum visibility decomposition of a polygon with $h \geq 0$ holes has only $\mathcal{O}((h+1)n^3)$ cells (Theorem 1) and only $\mathcal{O}((h+1)^2n^2)$ cells of minimal visibility (Theorem 2). We exploit this result to improve the running time of Ghosh’s algorithm from $\mathcal{O}(n^5)$ to $\mathcal{O}((h+1)^2n^3)$; this is a strict asymptotic improvement for $h = o(n)$.

Having presented algorithms with faster running times, we turn our attention to improving the approximation factor. We show that, with the same time complexity bound of $\mathcal{O}((h+1)^2n^3)$, we can apply standard random sampling techniques for range spaces of bounded VC dimension to obtain an approximation factor of $\mathcal{O}((1 + \log(h+1)) \log \text{OPT})$ for polygons with $h \geq 0$ holes (Theorem 5).

Finally, we show that, with the same time bound of $\mathcal{O}(n^3)$, we can achieve an approximation ratio of $\mathcal{O}(\log \log \text{OPT})$ for simple polygons using the improved ε -net finders of King and Kirkpatrick [19] (Theorem 6).

The author originally suggested the idea of exploiting the number of sinks to improve running time in a recent thesis [18]. This idea was also used independently by Jang and Kwon [17], who obtained the same time complexity as us for simple polygons. Jang and Kwon also consider the problem of edge guards, whereas we do not. However, they do not consider improvement of the $\mathcal{O}(\log n)$ approximation factor or consider polygons with holes.

1.5 Model of Computation

We assume the real-RAM model of computation [24]. We also assume that the polygon and any holes are non-degenerate, and that vertices of the polygon are in general position, *i.e.*, no three are collinear.

2 Polygon Decompositions

2.1 Notation

For a polygon P with $h \geq 0$ holes, $\bar{P} \cup \partial P$ is the complement of the relative interior of P , or the closure of the complement of P . We use C_0, \dots, C_h to denote the $h+1$ components of $\bar{P} \cup \partial P$. C_0 is the polygon’s exterior and C_1, \dots, C_h are the holes of the polygon.

In addition to left windows, right windows, left pockets, and right pockets defined by Bose *et al.* for simple polygons [5] (see Figure 3), we have T-windows,

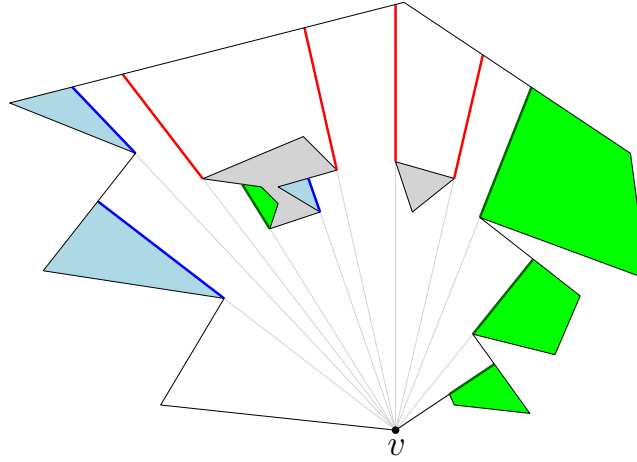


Fig. 3. Windows and pockets of a vertex v . Windows are represented by coloured line segments: left windows are blue, right windows are green, and T-windows are red. Left and right pockets are shaded blue and green respectively.

which are trans-component windows. A window is a T-window if its two endpoints are on different components. There are no pockets associated with T-windows.

A window of a vertex v has two endpoints; the endpoint closer to v is called the *base* and the endpoint farther from v is called the *end*. The *window half plane* of a left (resp. right) window of v whose base is b is the half plane on the left (resp. right) side of the oriented line from v to b .

2.2 Bounding the Number of Cells

Here we extend the results of Bose *et al.* [5] concerning the number of cells in a visibility decomposition. Our proofs are, to a great degree, extensions of their proofs.

Lemma 1 ([5]). *No point z in a pocket Q of a window w is visible to any point y that is inside w 's half plane but outside Q .*

Proof. The proof of [5, Lemma 1] works without modification when generalizing to polygons with holes. For completeness we restate it here.

Let w have b as its base and e as its end. Since $Q \cup w$ is a polygon, the line segment \overline{zy} intersects $Q \cup w$ at least once. As \overline{yz} cannot intersect \overline{be} , it intersects some other line segment on the boundary of the polygon. So, y and z are not visible with respect to the chain Q .

Lemma 2. *For any vertex v there are at most $2h$ T-windows of v .*

Proof. Consider the *component sequence* of a vertex v obtained as follows. Let r be a ray emanating from v that bisects the external angle at v (or, if v is a vertex of a hole, the angle interior to the hole). From its starting position, the r rotates clockwise around v , making one full rotation. Now consider the first component that is hit by r . As r rotates, the component hit changes exactly where there is a T -window of v . The changing sequence of components hit by r through one full rotation is the *component sequence* of v . This sequence starts and ends with P_E .

It remains to show that the component sequence of v is a Davenport-Schinzel sequence of order 2; since it is a sequence over $h+1$ symbols, this implies that the sequence length is at most $2h+1$ [26], which in turn implies that there are at most $2h$ T -windows of v . A Davenport-Schinzel sequence of order 2 is a sequence in which, for any two symbols a and b , the subsequence $\dots, a, \dots, b, \dots, a, \dots, b, \dots$ does not appear.

The rest of the proof follows a simple geometric argument illustrated in Figure 4. Assume that, in the rotation of r , it hits C_i at point u_1 , then hits C_j , then hits C_i at point u_2 . Then the union of C_i and line segments $\overline{vu_1}$ and $\overline{vu_2}$, there is a Jordan curve that contains the relative interior of C_j in its interior region. Thus C_j will not be hit by r in its rotation after it passes u_2 . \square

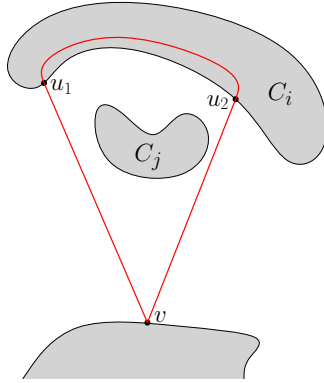


Fig. 4. If, in rotational order, v sees C_i , then C_j , then C_i again, then it will never see C_j after that because C_j is ‘trapped’ in the interior of the red Jordan curve.

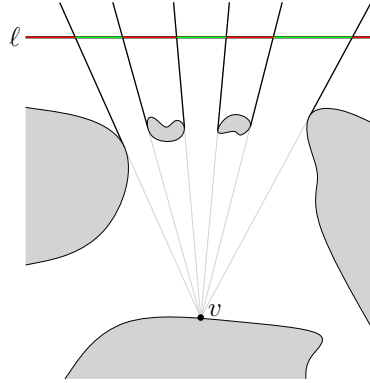


Fig. 5. An illustration accompanying the proof of Lemma 3. The line segment ℓ is intersected by 6 windows of v . Alternating intervals of visibility from v are shown in red and green. For ℓ to be intersected by more windows of v , more than 2 holes would be required.

Lemma 3. For a polygon P with h holes, a fixed vertex v , and a fixed line segment ℓ in P , ℓ intersects at most $2(h+1)$ windows of v : one right window, one left window, and $2h$ T -windows.

Proof. See Figure 5. Of the windows of p that cross ℓ , at most one is a right window and at most one is a left window. To see this, consider a point p moving from one endpoint of ℓ to the other, directed so that line segment \overline{vp} rotates clockwise around v as p moves. p can exit at most one left pocket of v and cannot enter a left pocket of v . p can enter at most one right pocket of v and cannot exit a right pocket of v . At most $2h$ T-windows of p cross ℓ because, by Lemma 2, there are at most $2h$ T-windows of p . \square

Theorem 1. *For a polygon P with h holes, $\mathcal{D}_V(P, V(P))$ contains $\mathcal{O}((h+1)n^3)$ cells.*

Proof. By Lemma 3, each window is crossed by at most $2(h+1)$ windows of each vertex, and thus $2(h+1)n$ other windows in total. The total number of windows is $\mathcal{O}(n^2)$, so the number of points at which windows cross is $\mathcal{O}((h+1)n^3)$. The theorem follows by applying Euler's formula for planar graphs. \square

2.3 Constructing and Using the Decomposition

Chazelle and Edelsbrunner [7] give an efficient algorithm for finding the planar decomposition defined by a set \mathcal{L} of line segments. The running time is $\mathcal{O}(|\mathcal{L}| \log |\mathcal{L}| + k)$, where k is the number of intersections. In our case, $|\mathcal{L}|$ is $\mathcal{O}(n^2)$ and Theorem 1 tells us that k is $\mathcal{O}((h+1)n^3)$, so the total runtime for this step is $\mathcal{O}((h+1)n^3)$.

The visibility decomposition $\mathcal{D}_V(P, V(P))$ has an associated planar graph G . We are particularly interested in a customized graph dual G' of G that we can use to find cells of minimal visibility. To obtain G' , we first build the dual graph of G . We then remove all edges corresponding to segments of ∂P . This gives us the underlying graph of G' , and we will direct the edges as follows.

Each edge of G that does not correspond to part of ∂P separates two cells; thus each edge of G' connects two cells. Consider an edge e of G' that connects two adjacent cells, or faces, f_i and f_j . If V_i and V_j are the respective sets of vertices that see points in f_i and f_j , then either V_i can be obtained from V_j by removing one vertex or vice versa. We direct e towards the face with the smaller corresponding set, *i.e.*, towards the cell that is less visible. Doing this for all edges we obtain a directed acyclic graph. The sinks of this graph correspond to the cells of minimal visibility in $\mathcal{D}_V(P, V(P))$.

Bose *et al.* used this technique and showed that, given the planar decomposition of a simple polygon P , in $\mathcal{O}(n^3)$ time it is possible to construct G' and identify all sinks. This construction extends naturally to polygons with holes in time $\mathcal{O}((h+1)n^3)$.

It is worth noting that Bose *et al.* used the planar decomposition algorithm of Bentley and Ottmann [3], which in this application is slower by a factor of $\log n$. They do this because they claim the algorithm of Chazelle and Edelsbrunner [7] requires the line segments to be in general position (they will not necessarily be in general position even if the polygon's vertices are in general position). However, we have looked for and failed to find any original mention of this general position

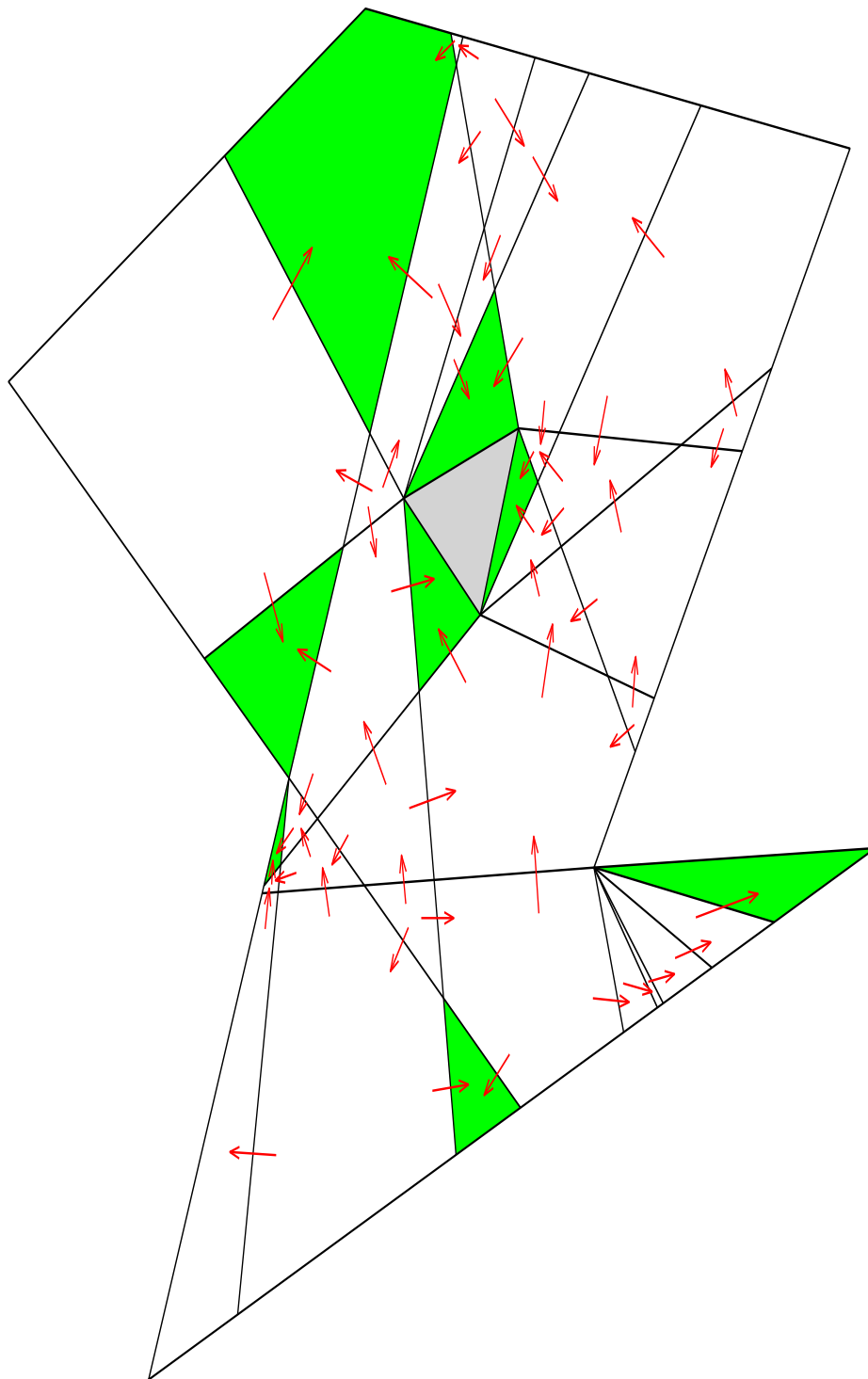


Fig. 6. The visibility decomposition of a polygon with its directed dual edges indicated in red and its sinks, *i.e.*, cells of minimal visibility, shaded green.

requirement. At any rate, degeneracies will only occur at the vertices of P , and an efficient workaround is possible.

2.4 Bounding the Number of Sinks

Here we give an upper bound of $\mathcal{O}((h+1)^2n^2)$ on the number of sinks. This generalizes the $\mathcal{O}(n^2)$ bound given by Bose *et al.* for simple polygons.

Lemma 4 ([5]). *Given two right pockets p_1 and p_2 , of a point x , no point in p_1 can see a point in p_2 .*

Proof. The proof of [5, Lemma 8] works without modification when generalizing to polygons with holes. For completeness we restate it here.

If a point a in p_1 could see a point b in p_2 , then the line segment between them must intersect both the window of p_1 and p_2 . Consider the window half planes of p_1 and p_2 . Since the two half planes intersect at x and both windows are right windows, one of the half planes must contain the windows of the other. The lemma follows from Lemma 1. \square

Lemma 5 ([5]). *There is at most one point of intersection between all the right windows of a point x and all the right windows of a distinct point y .*

Proof. The proof of [5, Lemma 9] works without modification when generalizing to polygons with holes. For completeness we restate it here.

Note that if a right window of x intersects a right window of y , then both x and y are visible from the intersection point. Also, from the fact that both are right windows, the base of one window must be contained in the pocket of the other. There are two cases to consider: either x is contained in a right pocket of y or x is not contained in a right pocket of y . We start with the former.

If x is contained in a right pocket of y , then by Lemma 4 x cannot see any other right window of y , and by Lemma 3, only one right window of x can intersect the right window of y 's right pocket containing x . Therefore, the lemma follows in this case.

Assume that x is not contained in a right pocket of y . Suppose that a right window r_1 of x intersects a right window r_2 of y . Since x is not contained in a right pocket of y , the base of r_2 must be contained in the pocket of r_1 . This implies that y is in the window half plane of r_1 . However, since y is visible from the intersection point of r_1 and r_2 , y must be in the pocket of r_1 by Lemma 1. Therefore, the lemma follows since y is contained in a right pocket of x . \square

Lemma 6. *For distinct vertices v_i and v_j , if v_i is not in a right pocket of v_j then any T -window of v_i crosses at most one right window of v_j .*

Proof. By Lemma 4, a ray shot from v_i cannot leave a right pocket of v_j and enter another right pocket of v_j . The lemma follows easily. \square

Corollary 1. *The total number of crossings between T -windows is $\mathcal{O}((h+1)^2n^2)$.*

Theorem 2. *For a polygon P with h holes, $\mathcal{D}_V(P, V(P))$ contains $\mathcal{O}((h+1)^2 n^2)$ cells of minimal visibility, and this is tight in the worst case.*

Proof. Let R_i be the set of right windows of v_i . We want to show that there are $\mathcal{O}((h+1)n)$ sinks bordered by a window in R_i . We do this by showing that, for a vertex v , there are $\mathcal{O}(h)$ sinks having an edge from a window in R_i , followed in clockwise order by an edge from a window of v . If v is in a right pocket of v_i , v can only see one window in R_i , namely the window bounding the pocket, and by Lemma 3 we are done. Otherwise, consider a sink s that has one edge formed by a right window $r \in R_i$ and the next edge in clockwise order formed by a window of v . The window of v must be either a right window or a T-window, otherwise s would not be a sink. By Lemma 5, at most one right window of v can intersect a window in R_i . By Lemma 6, each T-window of v crosses at most one window in R_i .

We have shown that there are $\mathcal{O}((h+1)n)$ sinks bordered by a window in R_i . We can do the same for sinks bordered by windows in L_i , the set of left windows of v_i . Thus the total number of sinks bordered by all right and left windows is $\mathcal{O}((h+1)n^2)$. By Corollary 1 we know that the number of cells having two consecutive sides formed by T-windows is $\mathcal{O}((h+1)^2 n^2)$, since each intersection point of two T-windows borders at most 4 cells. The upper bound of $\mathcal{O}((h+1)^2 n^2)$ for the total number of cells of minimal visibility follows.

We prove the lower bound by example. See Figures 7, 8, 9, and 10.

□

3 Greedy Approximation

Based on the previous section we treat the guarding problem as the abstracted problem of finding a minimum hitting set for a discrete range space $\mathcal{S} = (X, \mathcal{R})$, where $|X| = n$ and $|\mathcal{R}| = \mathcal{O}((h+1)^2 n^2)$. This range space can be obtained in time $\mathcal{O}((h+1)n^3)$.

Using the greedy algorithm for set cover/hitting set on this range space gives us a $\mathcal{O}(\log n)$ -approximation algorithm in $\mathcal{O}(|X||\mathcal{R}|) = \mathcal{O}((h+1)^2 n^3)$ time.

4 Improved Approximation via ϵ -Nets

We now turn our attention to the task of achieving a better approximation factor when OPT is small; for our bounds on time complexity we can assume that $\text{OPT} = \mathcal{O}(n^{1/3})$. Otherwise an approximation factor of $\mathcal{O}(\log n)$ is also $\mathcal{O}(\log \text{OPT})$.

4.1 VC-Dimension and ϵ -Nets

We use standard techniques for approximating hitting set that are based on the concept of *VC-dimension* first introduced by Vapnik and Chervonenkis [28] in the area of learning theory.

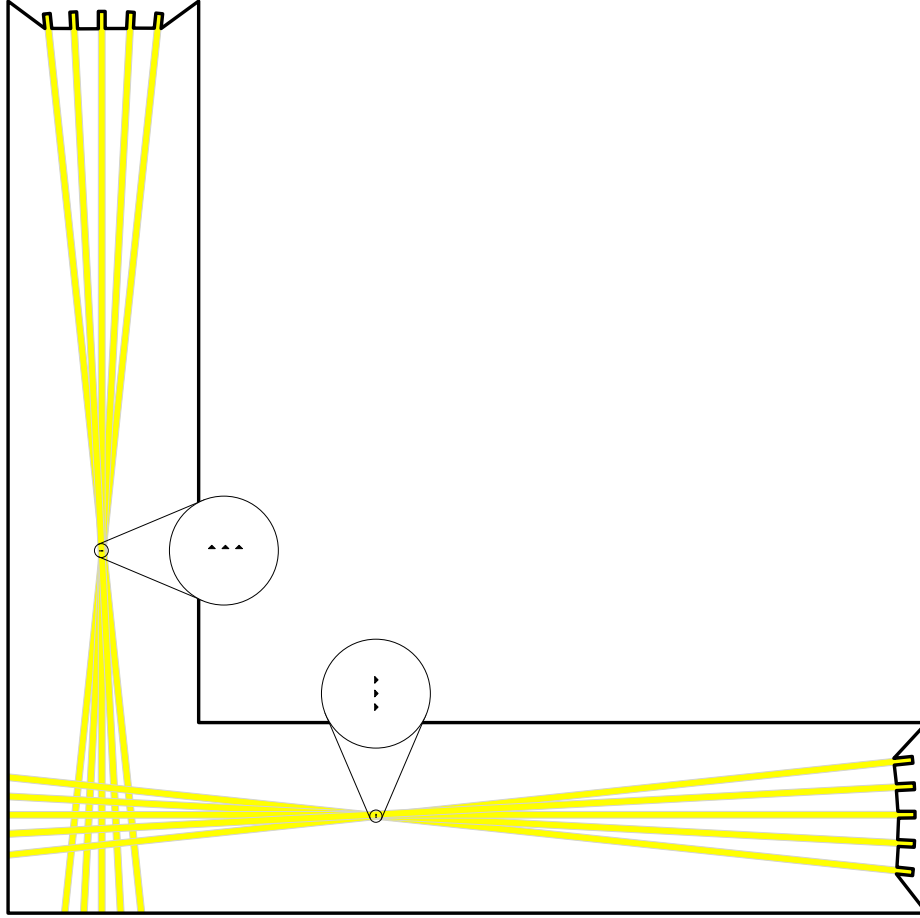


Fig. 7. A polygon with $\Theta((h+1)^2 n^2)$ sinks. Each *light strip*, indicated in yellow, is the intersection of visibility polygons of two adjacent vertices. This polygon has $10 = \Theta(n)$ light strips and 6 holes. The construction easily generalizes to higher values of n and h . This figure is diagrammatic; in the real polygon the holes would be closer to the region in which horizontal and vertical light strips interact.

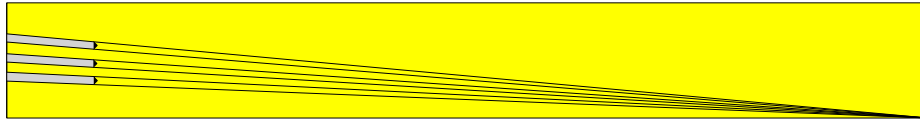


Fig. 8. Detail of one of the light strips. The *shadow strips*, indicated in light grey, are the regions not seen by the vertex at the bottom right. Each light strip has $h/2$ shadow strips.

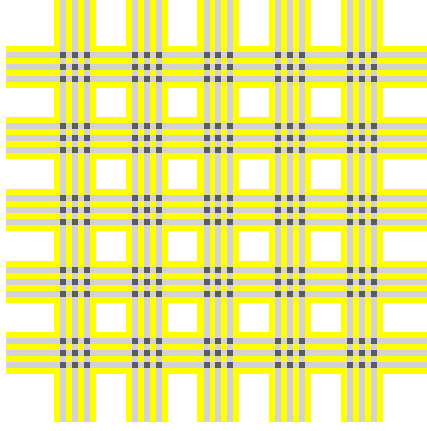


Fig. 9. Detail of the interaction of shadow strips. There are $\Theta((h+1)^2 n^2)$ dark grey squares, each of which is the intersection of shadow strips and must contain a sink.

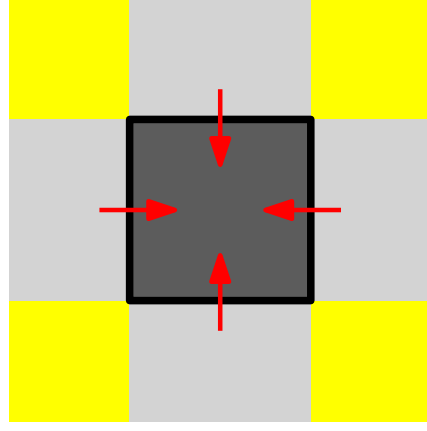


Fig. 10. Detail of the intersection of shadow strips. Dual edges are indicated in red; since they are all pointing inwards, the dark grey region must contain a sink.

Definition 1 (VC-Dimension [28]). For a range space $\mathcal{S} = (X, \mathcal{R})$, let Y be a maximum cardinality subset of X such that $\mathcal{R} \cap Y = 2^Y$. The VC-dimension of \mathcal{S} is equal to $|Y|$.

We use the following bounds on the VC-dimension of visibility systems in polygons due to Valtr [27].

Theorem 3 ([27]). The visibility system of a polygon with h holes has VC-dimension at most 23 if $h = 0$ (i.e., if the polygon is simple), and $2 \log_2 h + 4 \log_2 \log_2 h + o(1) = \mathcal{O}(1 + \log h)$ if $h \geq 1$.

In systems of bounded VC-dimension, small ε -nets can be constructed via random sampling. We use the following result of Blumer *et al.* [4].

Theorem 4 ([4]). For a measure μ on the elements of a range space of VC-dimension d , and for any $\varepsilon \in (0, 1]$ and any $\delta \in (0, 1]$, a random sample of $m(\varepsilon, \delta)$ elements drawn according to μ forms an ε -net with probability at least $1 - \delta$, where

$$m(\varepsilon, \delta) = \max \left(\frac{4}{\varepsilon} \log \frac{2}{\delta}, \frac{8d}{\varepsilon} \log \frac{13}{\varepsilon} \right).$$

The following is a straightforward consequence of Theorems 3 and 4.

Corollary 2. There exists a function $f(h, \varepsilon) = \mathcal{O} \left(\left(1 + \log(h+1) \right) \frac{1}{\varepsilon} \log \frac{1}{\varepsilon} \right)$ such that, for a measure μ on a polygon with $h \geq 0$ holes, and for any $\varepsilon \in (0, 1]$, a random sample of $f(h, \varepsilon)$ points drawn according to μ forms an ε -net with probability at least $1/2$.

4.2 Approximation via ε -Nets

Brönnimann and Goodrich [6] developed a method for turning algorithms for finding ε -nets into approximation algorithms for finding minimum hitting sets. The key is in finding an optimum, or approximately optimum, measure μ on the elements of a range space. Their algorithm essentially ‘learns’ a measure μ through an iterative doubling technique. The optimum measure μ^* is the distribution that maximizes the value of ε^* such that every ε^* -net is a hitting set. In fact, the size of a minimum fractional hitting set is exactly $\text{OPT}_f = 1/\varepsilon^*$ [12]; this serves as a lower bound for OPT .

Iterative doubling. The B&G algorithm [6, §3.1] finds a measure μ' such that every ε' -net is a hitting set for some $\varepsilon' \geq \frac{1}{2 \cdot \text{OPT}}$. The algorithm starts by assigning a weight of 1 to every element (when the algorithm terminates these weights are normalized to obtain μ'). In each iteration the algorithm makes one call to the ε -net finder, which returns an ε -net $Y \subseteq X$ and one call to a *verifier*. The verifier checks if the given set Y is a hitting set. If so, the algorithm returns Y and terminates. If not, the verifier returns a range R that is not hit by Y ; the algorithm doubles the weight of every element in R , then starts a new iteration. The algorithm is guaranteed to terminate after $\mathcal{O}(\text{OPT} \cdot \log |X|)$ iterations and the total weight of elements cannot exceed $|X|^4$ [6]. Since the B&G algorithm does not know the value of OPT or ε' *a priori*, it must make several guesses, starting at a constant value such as $1/2$ and halving the guess after each failed run of the algorithm. The result of this is that the entire algorithm must be run $\mathcal{O}(\log \text{OPT})$ times [6].

Random sampling and verification. We show that, for our application, the total time complexity of random sampling and verification is $\mathcal{O}(n^3)$.

For verification we build a directed bipartite graph with n vertices representing elements and $\mathcal{O}(n^2)$ elements representing ranges, where an element vertex is adjacent to a range vertex if and only if the element is in the range. This graph is constructed in $\mathcal{O}(n^3)$ time and can be constructed once and used for all iterations of the algorithm. Each range vertex is given a boolean flag indicating whether or not it is hit by the input set. In each verification round we do the following:

1. In $\mathcal{O}(|\mathcal{R}|)$ time, reset the flags.
2. In $\mathcal{O}(|Y| \cdot |\mathcal{R}|)$ total time, for each element in the input set Y , mark all incident ranges as ‘hit’.
3. In $\mathcal{O}(|\mathcal{R}|)$ time, scan through the ranges to find an unhit range or verify that all ranges are hit.
4. In $\mathcal{O}(|\mathcal{R}|)$ time, if there is an unhit range, double the weight of every element in that range.

Thus each verification round takes $\mathcal{O}(|Y| \cdot |\mathcal{R}|)$ time.

For random sampling, we assume that we can sample a random bit in $\mathcal{O}(1)$ time, allowing us to sample uniformly from an array of k elements in $\mathcal{O}(\log k)$ expected time. Through all iterations of the algorithm, every element weight is a

power of 2. Since the total element weight does not exceed $|X|^4$, each element has its weight doubled $\mathcal{O}(\log |X|)$ times in each run of the B&G algorithm. We can maintain a partition of the elements based on how many times each element's weight has been doubled. In this way we can sample an element according to the weight function in $\mathcal{O}(\text{polylog } |X|)$ amortized time. The additional cost of maintaining this partition is $\mathcal{O}(|X| \log |X|)$ time per run of the algorithm.

Total time complexity. For our application, we have that $Y = \mathcal{O}(n^{1/3} \text{polylog } n)$ and $|\mathcal{R}| = \mathcal{O}((h+1)^2 n^2)$. In each iteration of the B&G algorithm, the time complexity of sampling for the ε -net finder is

$$\begin{aligned} f(h, \varepsilon) \mathcal{O}(\text{polylog } n) &= \mathcal{O}\left(\left(1 + \log(h+1)\right) \frac{1}{\varepsilon} \log \frac{1}{\varepsilon} \text{polylog } n\right) \\ &= \mathcal{O}(\text{OPT} \cdot \log \text{OPT} \cdot \text{polylog } n) \\ &= \mathcal{O}\left(n^{1/3} \text{polylog } n\right). \end{aligned}$$

The time complexity of the verifier is $\mathcal{O}(|Y| \cdot |\mathcal{R}|) = \mathcal{O}((h+1)^2 n^{7/3} \text{polylog } n)$. We perform $\mathcal{O}(\log \text{OPT})$ runs of the algorithm and in each run there are $\mathcal{O}(\text{OPT} \cdot \log n)$ iterations. Thus the total number of iterations is $\mathcal{O}(\text{OPT} \text{polylog } n) = \mathcal{O}(n^{1/3} \text{polylog } n)$. Our total running time, including all runs of the algorithm, all iterations of the ε -net finder and verifier, and additional overhead, does not exceed $\mathcal{O}((h+1)^2 n^3)$.

Approximation ratio. When the last run of the algorithm terminates with success, we are left with a measure μ' on X , along with an ε' -net that is a hitting set, where $\varepsilon' \geq \frac{1}{2 \cdot \text{OPT}}$. The size of this hitting set is at most

$$\begin{aligned} f(h, \varepsilon') &= \mathcal{O}\left(\left(1 + \log(h+1)\right) \frac{1}{\varepsilon'} \log \frac{1}{\varepsilon'}\right) \\ &= \mathcal{O}\left(\left(1 + \log(h+1)\right) \text{OPT} \cdot \log \text{OPT}\right). \end{aligned}$$

Thus we have shown the following theorem.

Theorem 5. *For simple polygons or polygons with the number of holes bounded by a constant, there exists an approximation algorithm running in $\mathcal{O}(n^3)$ time with an approximation ratio of $\mathcal{O}(\log \text{OPT})$. For polygons with $h \geq 2$ holes, there exists an approximation algorithm running in $\mathcal{O}(h^2 n^3)$ time with an approximation ratio of $\mathcal{O}(\log h \log \text{OPT})$.*

5 Further Improved Approximation for Simple Polygons

For simple polygons, King and Kirkpatrick [19] presented an ε -net finder that returns ε -nets of size $\mathcal{O}\left(\frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon}\right)$; they did not analyze its time complexity, rather they simply stated that it runs in polynomial time. Here we show that it

runs in time $\mathcal{O}(n^2 \log \log \frac{1}{\varepsilon})$. This means that, for simple polygons, we can use it to replace the random sampling ε -net finder of the previous section without pushing the total running time above $\mathcal{O}(n^3)$.

First we note that the B&G algorithm always uses values of ε that are powers of 2. This means that

$$2^{\lceil \log \log 1/\varepsilon \rceil} = \mathcal{O}(1/\varepsilon) ,$$

which slightly facilitates analysis. With this restriction on ε , the K&K net finder recursively partitions the vertex set of the polygon, keeping its cyclic ordering. Define $t = \lceil \log \log t \rceil$. At the i^{th} level of the partition, each subset is further divided into b_i subsets, with

$$b_i = \begin{cases} 2^{2^{t-1}+1} \cdot 4t \cdot 2^{1-t} , & i = 1 \\ 2^{2^{t-i}+1} , & 1 < i \leq t . \end{cases}$$

If f_i is the number of new fragments created by the i^{th} fragmentation step, this gives us

$$f_i = \begin{cases} 1 , & i = 0 \\ 4t \cdot 2^{2^t - 2^{t-i} - t + i + 1} , & 0 < i \leq t \\ 4t \cdot 2^{2^t} , & i = t . \end{cases}$$

In $\mathcal{O}(n \log \log \frac{1}{\varepsilon})$ time we can build a tree corresponding to the hierarchical decomposition in which each node stores the corresponding subset in cyclic order. This tree has $\mathcal{O}(\frac{1}{\varepsilon} \log \log \frac{1}{\varepsilon})$ nodes. Determining guards to place for a pair of sibling fragments, respectively storing vertex sets U_1 and U_2 , can be done with $\mathcal{O}(|U_1||U_2|)$ calls to a visibility oracle. The visibility matrix of the vertices of P can easily be built in $\mathcal{O}(n^3)$ time and serves as a constant-time oracle.

Each of the f_{i-1} nodes at level $i-1$ contains $\mathcal{O}(n/f_{i-1})$ vertices; it has b_i normal children, each with $\mathcal{O}(n/f_i)$ vertices, and one dummy child with $\mathcal{O}(n)$ vertices. The cost of placing all guards at level i is therefore

$$\begin{aligned} \mathcal{O}\left(f_{i-1} \left(b_i^2 \left(\frac{n}{f_i} \right)^2 + b_i \left(n \cdot \frac{n}{f_i} \right) \right)\right) &= \mathcal{O}\left(\frac{n^2 b_i f_{i-1}}{f_i} \left(\frac{b_i}{f_i} + 1 \right)\right) \\ &= \mathcal{O}\left(n^2 \left(\frac{b_i}{f_i} + 1 \right)\right) \\ &= \mathcal{O}(n^2) . \end{aligned}$$

Thus the cost of placing all guards at all levels is $\mathcal{O}(n^2 \log \log \frac{1}{\varepsilon})$, as desired.

Using this ε -net finder instead of random sampling, we can achieve an approximation ratio of $\mathcal{O}(\log \log \text{OPT})$ for simple polygons in $\mathcal{O}(n^3)$ time.

Theorem 6. *For simple polygons, there exists an approximation algorithm running in $\mathcal{O}(n^3)$ time with an approximation ratio of $\mathcal{O}(\log \log \text{OPT})$.*

References

1. Aggarwal, A.: The Art Gallery Theorem: Its Variations, Applications and Algorithmic Aspects. Ph.D. thesis, The Johns Hopkins University (1984)
2. Alon, N., Moshkovitz, D., Safra, S.: Algorithmic construction of sets for k -restrictions. *ACM Transactions on Algorithms (TALG)* 2(2), 177 (2006)
3. Bentley, J.L., Ottmann, T.A.: Algorithms for reporting and counting geometric intersections. *Computers, IEEE Transactions on* 100(9), 643–647 (1979)
4. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM* 36(4), 929–965 (1989)
5. Bose, P., Lubiw, A., Munro, J.I.: Efficient visibility queries in simple polygons. *Computational Geometry: Theory and Applications* 23(3), 313–335 (2002)
6. Brönnimann, H., Goodrich, M.T.: Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry* 14(1), 463–479 (1995)
7. Chazelle, B., Edelsbrunner, H.: An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM (JACM)* 39(1), 1–54 (1992)
8. Deshpande, A., Kim, T., Demaine, E.D., Sarma, S.E.: A pseudopolynomial time $O(\log n)$ -approximation algorithm for art gallery problems. *Lecture Notes in Computer Science* 4619, 163–174 (2007)
9. Efrat, A., Har-Peled, S.: Guarding galleries and terrains. *Information Processing Letters* 100(6), 238–245 (2006)
10. Eidenbenz, S.: Inapproximability results for guarding polygons without holes. *Lecture Notes in Computer Science* 1533, 427–436 (1998)
11. Eidenbenz, S., Stamm, C., Widmayer, P.: Inapproximability results for guarding polygons and terrains. *Algorithmica* 31(1), 79–113 (2001)
12. Even, G., Rawitz, D., Shahar, S.: Hitting sets when the VC-dimension is small. *Information Processing Letters* 95(2), 358–362 (2005)
13. Feige, U.: A threshold of $\ln n$ for approximating set cover. *Journal of the ACM* 45(4), 634–652 (1998)
14. Ghosh, S.K.: Approximation algorithms for art gallery problems. In: *Proceedings of the Canadian Information Processing Society Congress*. pp. 429–434 (1987)
15. Ghosh, S.K.: Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics* 158(6), 718–722 (2010)
16. Guibas, L., Motwani, R., Raghavan, P.: The Robot Localization Problem. *SIAM Journal on Computing* 26(4), 1120–1138 (1997)
17. Jang, D.S., Kwon, S.I.: Fast approximation algorithms for art gallery problems in simple polygons. *ArXiv preprints* abs/1101.1346 (2011)
18. King, J.: Guarding Problems and Geometric Split Trees. Ph.D. thesis, McGill University (2010)
19. King, J., Kirkpatrick, D.: Improved approximation for guarding simple galleries from the perimeter, submitted
20. King, J.: Fast vertex guarding for polygons. *ArXiv preprints* abs/1101.3297 (2011)
21. Lee, D., Lin, A.: Computational complexity of art gallery problems. *IEEE Transactions on Information Theory* 32(2), 276–282 (1986)
22. O’Rourke, J.: *Art Gallery Theorems and Algorithms*. Oxford University Press (1987), <http://maven.smith.edu/~orourke/books/ArtGalleryTheorems/art.html>
23. O’Rourke, J., Supowit, K.J.: Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory* 29(2), 181–189 (1983)

24. Preparata, F.P., Shamos, M.I.: Computational Geometry: An Introduction. Springer Verlag, New York (1985)
25. Raz, R., Safra, S.: A sub-constant error-probability low-degree-test and a sub-constant error-probability PCP characterization of NP. In: Proceedings of the 29th ACM Symposium on Theory of Computing. pp. 475–484 (1997)
26. Sharir, M., Agarwal, P.K.: Davenport-Schinzel Sequences and Their Geometric Applications. Cambridge University Press (1995)
27. Valtr, P.: Guarding galleries where no point sees a small area. Israel Journal of Mathematics 104(1), 1–16 (1998)
28. Vapnik, V.N., Chervonenkis, A.Y.: On the uniform convergence of relative frequencies of events to their probabilities. In: Theory of Probability and its Applications. vol. 16, pp. 264–280 (1971)
29. Vazirani, V.V.: Approximation Algorithms. Springer Verlag, New York (2001)